MIT/LCS/TR-246

A SURVEY OF THE LOGIC OF EFFECTIVE DEFINITIONS

J. Tiuryn

A SURVEY OF THE LOGIC OF EFFECTIVE DEFINITIONS

by

J. Tiuryn
Institute of Mathematics,
Warsaw University,
Warsaw, Poland
and
Laboratory for Computer Science,
Massachusetts Institute of Technology,
Cambridge, Massachusetts 02139.

September 30, 1980.

Abstract: LED, the Logic of Effective Definitions, is an extension of first order predicate calculus used for making assertions about programs. Programs are modeled as effective definitional schemes (following Friedman). Logical properties of LED and its relations to classical logics and other programming logics are surveyed.

KEY WORDS: effective definitions, logic of programs, partial correctness, completeness, infinitary logic.

This work was supported in part by The National Science Foundation, Grant Nos. MCS 7719754 and MCS 8010707, and by a grant to the M.I.T. Laboratory for Computer Science by the IBM Corporation.

Contents

0. *Introduction*

The aim of this paper is to report on the latest version of the Logic of Effective Definitions. One of the main reasons for introducing this logic of programs comes from the fact that many existing logics of programs (cf. [1, 8, 10, 15, 21, 24, 25, 26, 27, 29]) are based on a fixed class of structured programs, e.g. flow-chart schemes or recursive procedures. This leads to some general questions such as:

(1) what properties of a given class of programs impact on the behaviors of the resulting logic?
(2) what are the limitations on the expressive power of logics of programs?
(3) what properties such as compactness, interpolation, etc. hold for logics of programs?
(4) what common methods might be applied in different logics of programs?

All these questions are difficult to answer at once. It is the author's opinion that there should exist a common framework in which all these questions can be embedded with a hope of getting some answers. This is the intended role of the Logic of Effective Definitions, LED.

LED is based on completely unstructured schemes which are better called *effective definitions* rather than programs. The only primitive relation in LED is *total equivalence* between schemes -- many other interesting notions are derivable from (expressible using) the primitive ones. The extremely simple structure of effective definitions together with the simplicity of LED formulas make model-theoretic methods easier to apply when attacking problems (1) - (4). On the other hand, many logics of programs can be retrieved as fragments of LED (cf. Section 5) via the standard *unfolding procedure* applied to the programs on which the logic is based.

We emphasize here that throughout this paper we consider only deterministic programs. There are no problems in formulating a non-deterministic version of LED. However, there are confusingly many open questions concerning deterministic programs and their logics. This situation suggests, in the author's opinion, a need for better understanding of the phenomena arising in the deterministic case before passing to nondeterminism.

The results presented in this paper are mainly concerned with LED itself. However, the open problems formulated in Section 5 are oriented towards a better understanding of the behavior of LED fragments.

To keep the paper a reasonable size, we give only brief sketches of proofs of results which appear elsewhere. Actually, there are three new results stated in this paper (3.5.5, 4.2.6, and 4.3.6) -- they are mainly improvements of the earlier ones. In this case more complete proofs are given.

The first version of LED (in [31]) was formulated for a three-valued logic -- the third truth value in this logic corresponded to divergence. A completeness theorem for this logic is proved in [32]. A reformulation of LED based upon merely two truth values is given in [33]. We introduce LED in this paper in essentially the same way as in [33].

I would like to thank Professor Albert R. Meyer for many valuable conversations, for his fruitful comments on the earlier versions of LED, and for raising several interesting questions concerning partial correctness theories. I further thank Professor Meyer and Joseph Halpern for editing this paper.

I would also like to express my thanks to Professor Klaus Indermark and his colleagues at the Lehrstuhl für Informatik II of Rheinisch-Westfälische T.H. in Aachen for a very sympathetic and stimulating atmosphere during my stay there in 1978/79, when some of the ideas and results presented in this paper were formulated.

Finally, thanks are due to Cindy Martignetti of the M.I.T. Laboratory for Computer Science for typing the paper.

1. *Preliminary Notions and Definitions*

In this section we recall some basic notions and definitions from logic. We concentrate here mainly on notation rather than on complete definitions of standard concepts — the latter can be found in any text on mathematical logic (e.g. [2, 7]).

1.1 Let ω denote the first infinite ordinal. As a set ω is equal to the set of all finite ordinals $\{0, 1, 2, \ldots\}$. We use ω^+ to denote the set $\omega - \{0\}$. A finite ordinal $n \in \omega$ is identified with the set of all ordinals smaller than n.

Let $\xi \leq \omega$ be an ordinal and let A be a set. Elements of A^ξ are called *ξ-vectors* over A, and they are functions from ξ into A. For every $a \in A^\xi$ and for every $n < \xi$, a_n is the n-th component of a, i.e. $a_n = a(n)$.

Where it does not lead to confusion we identify, for $n, k < \omega$, the sets $A^n \times A^k$ and A^{n+k}.

1.2 By a *language* L we mean an ordered pair $L = \langle L, \rho_L \rangle$, where $L = L_C \cup L_F \cup L_R$ is a union of pairwise disjoint sets L_C, L_F, L_R called the set of *constant* symbols, *function* symbols, and *predicate* symbols, respectively and $\rho_L : L_F \cup L_R \to \omega^+$ is a function called the *arity function.*

1.3 Let L be a language and let $X = \{x_n : n < \omega\}$ be a set disjoint from L. The set X will be fixed throughout the paper. Elements of X are called *individual variables.*

Let T(L) denote the set of all terms of L with variables from X, and $L_{\omega\omega}(L)$ the set of all first-order formulas over L augmented by the equality symbol with variables from X. Finally, let OF(L) denote the set of all open (i.e. quantifier-free) formulas from $L_{\omega\omega}(L)$.

For $t \in T(L)$, Var(t) is the set of all variables which occur in t. For $\alpha \in L_{\omega\omega}(L)$, Var($\alpha$) is the set of all variables which occur free in α. For every $n < \omega$ we define

$$T(L, n) = \{t \in T(L) : \mathrm{Var}(t) \subseteq \{x_i : i < n\}\},$$

$$L_{\omega\omega}(L, n) = \{\alpha \in L_{\omega\omega}(L) : \mathrm{Var}(\alpha) \subseteq \{x_i : i < n\}\}, \text{ and}$$

$$OF(L, n) = OF(L) \cap L_{\omega\omega}(L, n).$$

The elements of $L_{\omega\omega}(L, 0)$ are called *sentences*.

For $t \in T(L)$, *arity*(t) is the least $n < \omega$ such that $t \in T(L, n)$. Similarly, for $\alpha \in L_{\omega\omega}(L)$, arity$(\alpha)$ is the least $n < \omega$ such that $\alpha \in L_{\omega\omega}(L, n)$.

1.4 Let L be a language. By an L-*structure* $\mathfrak{A}$ we mean a set A called the *carrier* of $\mathfrak{A}$, and an interpretation of symbols in L (i.e. a function $s \to s^{\mathfrak{A}}$, for $s \in L$) which satisfies the following conditions:

1.4.1 if $c \in L_C$ then $c^{\mathfrak{A}} \in A$;

1.4.2 if $f \in L_F$ and $\rho_L(f) = n$, then $f^{\mathfrak{A}}: A^n \to A$;

1.4.3 if $r \in L_R$ and $\rho_L(r) = n$, then $r^{\mathfrak{A}} \subseteq A^n$.

An arbitrary $t \in T(L)$ determines in an L-structure $\mathfrak{A}$ a function $t^{\mathfrak{A}} : A^{\omega} \to A$ which is defined inductively in the obvious way ($t^{\mathfrak{A}}$ is said to be the *meaning* of t in $\mathfrak{A}$). The value of this function on a given $a \in A^{\omega}$ depends only on the first arity(t) components of a. Therefore we shall sometimes write ambiguously $t^{\mathfrak{A}}(a)$, where $a \in A^k$ and arity$(t) \leq k$, viewing $t^{\mathfrak{A}}(a)$ as the value of $t^{\mathfrak{A}}$ on any extension of a to an ω-vector over A.

For an L-structure $\mathfrak{A}$, $a \in A^{\omega}$, and $\alpha \in L_{\omega\omega}(L)$, $\langle\mathfrak{A}, a\rangle \models \alpha$ means α is true of $\mathfrak{A}$ under the valuation of variables a. Just as for terms, the truth of α in $\langle\mathfrak{A}, a\rangle$ depends only on the first n components of a, where n = arity(α). For arity$(\alpha) \leq k$ and $a \in A^k$, $\mathfrak{A} \models \alpha[a]$ means $\langle\mathfrak{A}, a^*\rangle \models \alpha$ for any extension a^* of a to an ω-vector over A.

We shall write $\mathfrak{A} \models \alpha$ if for every $a \in A^{\omega}$, $\langle\mathfrak{A}, a\rangle \models \alpha$. If $\mathfrak{A} \models \alpha$ then $\mathfrak{A}$ is said to be a *model* for α. We write $\models \alpha$ if for every L-structure $\mathfrak{A}$, $\mathfrak{A} \models \alpha$.

We extend the above definitions to sets of formulas. If $\Sigma \subseteq L_{\omega\omega}(L)$ and $\mathfrak{A}$ is an L-structure then we write $\mathfrak{A} \models \Sigma$ if for every

$\alpha \in \Sigma$, $\mathfrak{A} \models \alpha$ holds. If this is the case then $\mathfrak{A}$ is said to be a model for Σ. We write $\models \Sigma$ if every L-structure $\mathfrak{A}$ is a model for Σ.

Finally, if $\Sigma \subseteq L_{\omega\omega}(L)$ and $\alpha \in L_{\omega\omega}(L)$, then we write $\Sigma \models \alpha$ if every model for Σ is a model for α.

1.5 For every finite language L we adopt a standard *Gödel coding* for the expressions in T(L) and OF(L) (cf. for example [2]).

2. *Friedman's Effective Definitions*

The notion of *effective definition* is due to H. Friedman ([12]). In section 2.1 we will define *effective definitional schemes* over a finite language **L**. They will be semantically equivalent (in all total interpretations) to Friedman's effective definitions over **L** augmented by =, a binary predicate symbol which is always interpreted as equality. We defer until later a full discussion of the appropriateness of our definition, but one pragmatic motivation is that we want our Logic of Effective Definitions to be similar to Deterministic Dynamic Logic, where tests for equality are allowed. (cf. 5.2).

Friedman's effective definitions are known to be of universal (computational) power over total interpretations (cf. [30] for discussion and further references). Many other classes of program schemes, e.g. flowcharts with indexed variables ([30]) or flowcharts with a stack and counters [23], are inter-translatable with the class of effective definitions. This phenomenon provides a system of finite descriptions which is semantically equivalent to effective definitions, the latter being infinite objects. We have decided *not* to introduce finitary descriptions since they tend to be distracting. For example, many of our proofs involve constructing a new scheme from a given one. This construction is often easily described in English, but a formal description of the construct tends to be complex. Since our entire development depends only on the schemes involved and not on how they are described, there is certainly no harm in omitting such a system of finite descriptions.

2.1 *Effective Definitional Schemes*

Let **L** be a finite language and let $n \in \omega$. By an *effective definitional scheme* (eds) S (over **L**) with variables among $\{x_i : i < n\}$ we mean a recursive function $S : \omega \to OF(\mathbf{L}, n) \times T(\mathbf{L}, n)$ (S is recursive with respect to the codings fixed in 1.5). The set of all effective definitional schemes over **L** with variables in $\{x_i : i < n\}$ is denoted by $ED(\mathbf{L}, n)$. The set of all eds's over **L** is denoted by $ED(\mathbf{L})$ and is equal to $\bigcup_{n\in\omega} ED(\mathbf{L}, n)$.

We adopt the following useful notation. If $S \in ED(\mathbf{L})$ and $m \in \omega$, then $\alpha_{S,m}$ is the first component and $t_{S,m}$ is the second component of the pair $S(m)$, i.e. $S(m) = \langle \alpha_{S,m}, t_{S,m} \rangle$. For $S \in ED(\mathbf{L})$ we define arity(S) to be the least $n < \omega$ such that $S \in ED(\mathbf{L}, n)$.

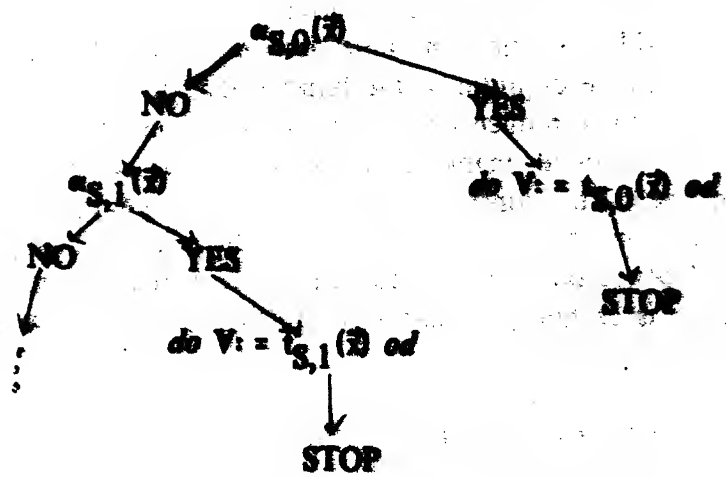
Let $\mathfrak{A}$ be an L-structure and let $S \in ED(L, n)$ for some $n \in \omega$. The scheme S defines in $\mathfrak{A}$ a partial function $S^{\mathfrak{A}} : A^n \to A$, which is defined in the following way:

$$S^{\mathfrak{A}}(a) = t^{\mathfrak{A}}_{S,i}(a) \quad \text{where } i < \omega \text{ is the least element in the set } \{k < \omega : \mathfrak{A} \models \alpha_{S,k}[a]\};$$

$$\text{undefined, if there is no such } i.$$

We write $S^{\mathfrak{A}}(a)\downarrow$ to indicate that $S^{\mathfrak{A}}$ is defined at a.

From the above definition we see that effective definitional schemes are schemes of definitions by cases (i.e. by a recursively enumerable set of cases). An eds S can be viewed as the following infinite conditional:



with V being the output variable.

Just as in 1.4, we will slightly abuse the notation and write $S^{\mathfrak{A}}(a)$, where $a \in A^k$ and $\text{arity}(S) \leq k$. This should not lead to confusion, since the result $S^{\mathfrak{A}}$ on a depends on, at most, the first arity(S) components of a.

An eds $S \in ED(L, n)$ is said to be *deterministic* if for every L-structure $\mathfrak{A}$ and for every $a \in A^n$, the set $\{k \in \omega : \mathfrak{A} \models \alpha_{S,k}[a]\}$ has at most one element.

The next definition is an obvious generalization of the notion of an effective definitional scheme.

2.1.1 *Recursively Enumerable Tree-schemes*

Let L be a finite language and let $n \in \omega$. We describe here r.e. trees which compute n-ary functions in L-structures. (cf. [17]).

The *input variables* are $\{x_i : i < n\}$, there is one *output variable* z, and a countable set $\{v_i : i \in \omega\}$ of *auxiliary variables*. We assume that $z \notin \{x_i : i < n\} \cup \{v_i : i \in \omega\}$.

Test conditions are arbitrary first-order open formulas over L (with equality) with variables in $\{x_i : i < n\} \cup \{v_i : i \in \omega\}$.

Assignment statements are expressions of the form $y := t$, where $y \in \{x_i : i < n\} \cup \{v_i : i \in \omega\}$ and t is a term over L with variables in $\{x_i : i < n\} \cup \{v_i : i \in \omega\}$. The variable y is called the *left side expression* of the assignment $y := t$.

Halt statements are expressions of the form STOP($z := t$), where t is a term over L with variables in $\{x_i : i < n\} \cup \{v_i : i \in \omega\}$.

Consider countable rooted trees with the property that every vertex has at most two successors. Each vertex with two successors is labeled by a test condition, each vertex with exactly one successor is labeled by an assignment statement, and each leaf is labeled by a halt statement. Moreover we add a technical condition: for each path π leading from the root to a vertex labeled by a test condition α (resp. an assignment statement $y := t$ or a halt statement STOP($z := t$)) if an auxiliary variable v_i occurs in α (or in the term t in the case of assignment/halt statement), then there is a subpath π' of π leading from the root to a vertex labeled by an assignment statement with v_i on the left side.

Let T be a tree satisfying the above-mentioned conditions. For any path π in T let e_π be a formal concatenation of all expressions which label vertices on that path (in the order in which they occur). Call T a *recursively enumerable tree-scheme* if the set $\{\langle e_\pi, \pi\rangle : \pi$ leads in T from the root to a leaf$\}$ is a r.e. set.

Let $\mathfrak{A}$ be an L-structure and let T be a r.e. tree-scheme over L with input variables in $\{x_i : i < n\}$. The computation of T in $\mathfrak{A}$ for input value $a \in A^n$ is defined naturally. It starts with a being substituted for the input variables. Then the assignment statements are performed in the obvious way. If the computation reaches a test condition α (along a path π)

then the next instruction to be executed is the instruction labeling the vertex reached either by $\pi 0$ or by $\pi 1$, depending on whether or not the test α is false at this stage in the computation. When the computation reaches a halt statement then it stops with the output computed from the term on the right hand side of the statement. Let $T^{\mathfrak{A}} : A^n \to A$ be the partial function computed by T in $\mathfrak{A}$.

2.1.2 *Proposition*

Let L be a finite language and let $n \in \omega$.

(i) For every eds $S \in ED(L, n)$ there is a deterministic eds $Q \in ED(L, n)$ such that in every L-structure $\mathfrak{A}$, $S^{\mathfrak{A}} = Q^{\mathfrak{A}}$.

(ii) For every r.e. tree-scheme T over L with variables in $\{x_i : i < n\}$ there is an $S \in ED(L, n)$ such that in every L-structure $\mathfrak{A}$, $T^{\mathfrak{A}} = S^{\mathfrak{A}}$.

Proof: The proof of (i) is obvious. For (ii), suppose π is a path in T from the root to a halt statement. Let e_π be the corresponding concatenation of expressions occurring on that path. By executing a formal computation along the path π we produce a pair (α_π, t_π), where $\alpha_\pi \in OF(L, n)$ represents a logical history of that path, and $t_\pi \in T(L, n)$ represents a result for that path. The set of all pairs $\langle \alpha_\pi, t_\pi \rangle$, where π ranges over all terminating paths is an eds with the required properties. ∎

Let $S \in ED(L)$ and let $n < \omega$. Define an eds $S^{(n)} \in ED(L)$ by:

$$S^{(n)}(k) = \begin{cases} S(k) & \text{for } k < n \\ S(n) & \text{for } n \leq k \end{cases}$$

An eds $S \in ED(L)$ is said to be *finite* if for some $n < \omega$, S is equal to $S^{(n)}$. Finite eds's correspond to *straight-line* programs augmented by an ABORT or DIVERGE statement.

2.2 *Recursion-theoretic notions relative to a given structure*

Let L be a finite language, let $\mathfrak{A}$ be an L-structure, and let $n < \omega$. A partial function $f: A^n \to A$ is said to be $\mathfrak{A}$-*computable* if there is an eds $S \in ED(L, n)$ with $f = S^{\mathfrak{A}}$. A subset $W \subseteq A^n$ is said to be $\mathfrak{A}$-*semicomputable* if W is the domain of an $\mathfrak{A}$-computable function.

Let S, $Q \in ED(L, n)$, and let $\mathfrak{A}$ be a L-structure. Define a subset $(S^{\mathfrak{A}} \doteq Q^{\mathfrak{A}}) \subseteq A^n$, by

$(S^{\mathfrak{A}} \doteq Q^{\mathfrak{A}}) = \{a \in A^n : S^{\mathfrak{A}}(a)\downarrow, Q^{\mathfrak{A}}(a)\downarrow, \text{ and } S^{\mathfrak{A}}(a) = Q^{\mathfrak{A}}(a)\}$.

2.2.1 *Proposition* ([33])

Let L be a finite language and let $n < \omega$. Let S, $Q \in ED(L, n)$. Then there exist P_1, P_2, $P_3 \in ED(L, n)$ which can be effectively found from indices for S and Q, such that for every L-structure $\mathfrak{A}$ and for every $a \in A^n$,

(i) $a \in (S^{\mathfrak{A}} \doteq Q^{\mathfrak{A}})$ iff $P_1^{\mathfrak{A}}(a)\downarrow$;

(ii) $S^{\mathfrak{A}}(a)\downarrow$ and $Q^{\mathfrak{A}}(a)\downarrow$ iff $P_2^{\mathfrak{A}}(a)\downarrow$;

(iii) Either $S^{\mathfrak{A}}(a)\downarrow$ or $Q^{\mathfrak{A}}(a)\downarrow$ iff $P_3^{\mathfrak{A}}(a)\downarrow$.

Proof:

(i) By 2.1.2(i) we may assume that S and Q are deterministic. Let

$$() : \omega^2 \to \omega$$

be a pairing function (i.e. a recursive one-to-one mapping of ω^2 onto ω; cf. [28]. Then

$P_1((m, k)) = \langle \alpha_{S,m} \wedge \alpha_{Q,k} \wedge t_{S,m} = t_{Q,k}, t_{S,m} \rangle$, for $m, k < \omega$, is an eds with the required properties.

(ii) Again we may assume S and Q are deterministic. Let

$$P_2((m, k)) = (\alpha_{S,m} \wedge \alpha_{Q,k}, t_{S,0}).$$

Then clearly $P_2^{\mathfrak{A}}(a)\downarrow$ iff $S^{\mathfrak{A}}(a)\downarrow$ and $Q^{\mathfrak{A}}(a)\downarrow$.

(iii) Is obvious. ∎

2.2.2 *Corollary*

For arbitrary eds's S, Q and for an arbitrary L-structure $\mathfrak{A}$, $S^{\mathfrak{A}} \doteq Q^{\mathfrak{A}}$ is $\mathfrak{A}$-semicomputable. Moreover all $\mathfrak{A}$-semicomputable sets are closed under finite unions and intersections.

2.2.3 *Example*

Let $\mathfrak{N} = \langle \omega, S, 0 \rangle$ be a structure with a unary function S successor and a constant $0 \in \omega$. Then the $\mathfrak{N}$-computable functions are precisely the partial recursive functions.

3. *Logic of Effective Definitions*

3.1 *Syntax and Semantics*

Let L be a finite language. Let LED(L) be the least set of expressions satisfying 3.1.1 - 3.1.3 below. Elements of LED(L) are called *LED formulas*.

3.1.1 If S, Q $\in$ ED(L) then $S \doteq Q \in$ LED(L).

3.1.2 If α, $\beta \in$ LED(L) then $\neg\alpha$, $(\alpha \wedge \beta)$, and $(\alpha \vee \beta)$ belong to LED(L).

3.1.3 If $\alpha \in$ LED(L) and $x_n \in X$ is an individual variable then $\exists x_n \alpha$ and $\forall x_n \alpha$ belong to LED(L).

Open LED formulas form the least subset of LED(L) closed under 3.1.1 and 3.1.2. An open LED formula α is said to be *positive* if the negation sign ($\neg$) does not occur in α.

We introduce the following abbreviations for formulas:

$$\alpha \to \beta \text{ is used for } \neg\alpha \vee \beta$$
$$\alpha \leftrightarrow \beta \text{ is used for } (\alpha \to \beta) \wedge (\beta \to \alpha).$$

If $\alpha \in$ LED(L) and if $\mathfrak{A}$ is a L-structure and $a \in A^{\omega}$, then $\langle\mathfrak{A}, a\rangle \models \alpha$ means that "α is true in $\mathfrak{A}$ under valuation a" $\langle\mathfrak{A}, a\rangle \models \alpha$ is defined by induction on the complexity of α, as follows:

3.1.4 If α is $S \doteq Q$ where S, Q $\in$ ED(L) then

$$\langle\mathfrak{A}, a\rangle \models \alpha \text{ iff } a \in S^{\mathfrak{A}} \doteq Q^{\mathfrak{A}}.$$

3.1.5 If α is $\neg\beta$ then

$$\langle\mathfrak{A}, a\rangle \models \alpha \text{ iff } \textit{not} \langle\mathfrak{A}, a\rangle \models \beta.$$

3.1.6 If α is $\beta_1 \wedge \beta_2$ then

$\langle \mathfrak{A}, a \rangle \models \alpha$ iff both $\langle \mathfrak{A}, a \rangle \models \beta_1$ and $\langle \mathfrak{A}, a \rangle \models \beta_2$.

3.1.7 If α is $\beta_1 \vee \beta_2$ then

$\langle \mathfrak{A}, a \rangle \models \alpha$ iff either $\langle \mathfrak{A}, a \rangle \models \beta_1$ or $\langle \mathfrak{A}, a \rangle \models \beta_2$.

3.1.8 If α is $\forall x_n \beta$ then

$\langle \mathfrak{A}, a \rangle \models \alpha$ iff for all $a' \in A^\omega$ such that $a_k' = a_k$ for $k \neq n$, $\langle \mathfrak{A}, a' \rangle \models \beta$ holds.

3.1.9 If α is $\exists x_n \beta$ then $\langle \mathfrak{A}, a \rangle \models \alpha$ iff *not* $\langle \mathfrak{A}, a \rangle \models \forall x_n \neg \beta$.

Just as in the first order logic, we see that the truth of α in $\langle \mathfrak{A}, a \rangle$ depends only on the first n components $a_0, \ldots, a_{n-1}$ of a, where n is the greatest integer such that x_{n-1} occurs free in α. We set arity(α) = n, where n is defined above.

We write $\mathfrak{A} \models \alpha$ if for every $a \in A^\omega$, $\langle \mathfrak{A}, a \rangle \models \alpha$ holds; in this case $\mathfrak{A}$ is said to be a *model* for α. Finally we write $\models \alpha$ if for every L-structure $\mathfrak{A}$, $\mathfrak{A} \models \alpha$; in this case α is said to be a *tautology* of LED.

3.2 *Properties Expressible in LED*

3.2.1 *Total Equivalence*

It follows immediately from our definitions that for S, Q $\in$ ED(L), $\models S \doteq Q$ iff S and Q are totally equivalent (cf. [14]). Therefore LED can be viewed as a "first-order logic" built up from atomic formulas which express total equivalence of Friedman schemes.

3.2.2 *Strong Equivalence*

For S, Q $\in$ ED(L), let $S \equiv Q$ be an abbreviation of the LED formula $(S \doteq S \vee Q \doteq Q) \rightarrow S \doteq Q$. It is easy to check that for an L-structure $\mathfrak{A}$, $\mathfrak{A} \models S \equiv Q$ iff $S^{\mathfrak{A}} = Q^{\mathfrak{A}}$. Therefore $\models S \equiv Q$ iff S and Q are strongly equivalent (cf. [14]).

3.2.3 *Weak Equivalence*

For S, Q $\in$ ED(L), let S $\sim$ Q be an abbreviation of the LED formula $(S \doteq S \wedge Q \doteq Q) \rightarrow S \doteq Q$. If it easy to see that for an L-structure $\mathfrak{A}$, $\mathfrak{A} \models S \sim Q$ iff both $S^{\mathfrak{A}}$ and $Q^{\mathfrak{A}}$ can be extended to the same total function. Therefore $\models S \sim Q$ iff S and Q are weakly equivalent (cf. [14]).

3.2.4 *Termination Properties*

For an eds S $\in$ ED(L) it is easy to see that $S \doteq S$ expresses the property that S terminates, i.e. for an L-structure $\mathfrak{A}$, $\mathfrak{A} \models S \doteq S$ iff $S^{\mathfrak{A}}$ is total. In the sequel we use the more suggestive notation $S\downarrow$ for $S \doteq S$. We also write $S\uparrow$ for $\neg(S \doteq S)$.

3.2.5 *First-order Properties*

First-order logic $L_{\omega\omega}(L)$ is naturally interpretable in LED(L) in the following sense: for every $\alpha \in L_{\omega\omega}(L)$ there exists $\varphi_\alpha \in$ LED(L) which can be effectively found from (the code of) α such that for every L-structure $\mathfrak{A}$ and for every $a \in A^\omega$, $\langle\mathfrak{A}, a\rangle \models \alpha$ iff $\langle\mathfrak{A}, a\rangle \models \varphi_\alpha$.

For an open formula $\alpha \in$ OF(L) we define φ_α to be $S_\alpha \doteq S_\alpha$, where $S_\alpha \in$ ED(L) is a finite eds defined by $S_\alpha(k) = \langle\alpha, x_0\rangle$ for all $k < \omega$.

If $\alpha \in L_{\omega\omega}(L)$ is an arbitrary formula then first we take the *prenex normal form* of α, say $[-]\alpha^*$, where $[-]$ is a block of quantifiers and α^* is an open formula. Then we set φ_α to be $[-]\varphi_{\alpha^*}$.

Therefore we may assume that $L_{\omega\omega}(L)$ is included in LED(L), i.e. if a first-order formula α occurs as a subformula of an expression β which is intended to be a LED formula then α is viewed as the corresponding formula φ_α described above.

3.2.6 *Representation of Terms*

If $t \in$ T(L) then the finite eds $S_t \in$ ED(L) defined by $S_t(m) = \langle t{=}t, t\rangle$ for $m < \omega$, represents the term t, i.e. exactly the same individual variables occur in t and S_t, and for every L-structure $\mathfrak{A}$ and for every $a \in A^\omega$, $t^{\mathfrak{A}}(a) = S_t^{\mathfrak{A}}(a)$.

If $S \in ED(L)$ and $t_1, t_2 \in T(L)$ then $S \doteq t_1$, $t_1 \doteq S$, and $t_1 \doteq t_2$ are abbreviations of the LED formulas $S \doteq S_{t_1}$, $S_{t_1} \doteq S$, and $S_{t_1} \doteq S_{t_2}$.

3.2.7 *Partial Correctness*

Let L be a finite language, let $S \in ED(L, n)$, where $n < \omega$, and let $\alpha \in L_{\omega\omega}(L, n)$ and $\beta \in L_{\omega\omega}(L, n+1)$ be arbitrary first-order formulas. The LED formula $\alpha \rightarrow \forall x_n(S \doteq x_n \rightarrow \beta)$ expresses the *partial correctness* of S with respect to the input condition α and the output condition β (cf. [14]), i.e. for any L-structure $\mathfrak{A}$ and for $a \in A^\omega$, $\langle \mathfrak{A}, a\rangle \models \alpha \rightarrow \forall x_n(S \doteq x_n \rightarrow \beta)$ iff whenever $\mathfrak{A} \models \alpha[a_0, \ldots, a_{n-1}]$ and $S^{\mathfrak{A}}(a_0, \ldots, a_{n-1})$ terminates with result $b \in A$, then $\mathfrak{A} \models \beta[a_0, \ldots, a_{n-1}, b]$.

3.2.8 *Total Correctness*

For L, S, α, β as above (in 3.2.7) the LED formula $\alpha \rightarrow \exists x_n(S \doteq x_n \wedge \beta)$ expresses the *total correctness* of S with respect to the input condition α and the output condition β (cf. [14]), i.e. for any L-structure $\mathfrak{A}$ and for any $a \in A^\omega$, $\langle \mathfrak{A}, a\rangle \models \alpha \rightarrow \exists x_n(S \doteq x_n \wedge \beta)$ iff whenever $\mathfrak{A} \models \alpha[a_0, \ldots, a_{n-1}]$, then $S^{\mathfrak{A}}(a_0, \ldots, a_{n-1})$ terminates with result $b \in A$ and $\mathfrak{A} \models \beta[a_0, \ldots, a_{n-1}, b]$.

3.2.9 *Relation to* $L_{\omega_1\omega}$

For a finite language L, let $L_{\omega_1\omega}(L)$ denote the set of all formulas over L of the logic $L_{\omega_1\omega}$ (cf. [16]). $L_{\omega_1\omega}(L)$ differs from $L_{\omega\omega}(L)$ in that we allow countable conjunctions and disjunctions.

We now show that LED formulas are translatable into $L_{\omega_1\omega}$ formulas. The only thing we have to do is to show how to translate a formula of the form $S \doteq Q$, where S and Q are eds's, into an $L_{\omega_1\omega}$ formula. First, observe that $S \doteq Q$ is semantically equivalent (i.e. equivalent in all L-structures) to the infinite disjunction $\bigvee_{i,j<\omega} \varphi_{i,j}$, where $\varphi_{i,j} \in OP(L)$ expresses that S stops at the i-th step, Q stops at

the j-th step and both give the same result, i.e. $\varphi_{i,j}$ is:

$(\wedge_{m<i}\neg a_{S,m}) \wedge (\wedge_{k<j}\neg a_{Q,k}) \wedge a_{S,i} \wedge a_{Q,j} \wedge$
$t_{S,i} = t_{Q,j}$.

We shall make use of the observation that LED is interpretable in $L_{\omega_1\omega}$ in later sections.

3.3 *Normal-form Results for LED Formulas*

The following question arises naturally: What properties of programs are expressible by LED formulas? The results stated below give a partial answer to this question.

3.3.1 *Theorem* ([33])

Let L be a finite language.

(1) For every positive open formula $\alpha \in$ LED(L) there exists an eds $S \in$ ED(L), which can be found effectively from (the code of) α such that

$$\models \alpha \leftrightarrow S\downarrow$$

(2) For every open formula $\alpha \in$ LED(L) there exist $n < \omega$ and eds's S_i, $Q_i \in$ ED(L) for $i = i, \ldots, n$ such that n, $\{S_i : i < n\}$, and $\{Q_i : i < n\}$ can be effectively and found from α and

$$\models \alpha \leftrightarrow \wedge_{i<n}(S_i \equiv Q_i)$$

(3) For every open formula $\alpha \in$ LED(L) there exist $n < \omega$ and eds's S_i, $Q_i \in$ ED(L) for $i < n$ which can be effectively found from α such that

$$\models \alpha \leftrightarrow \wedge_{i<n}(S_i\downarrow \to Q_i\downarrow)$$

(4) For every open formula $\alpha \in$ LED(L) there exists a recursively enumerable set $\{S_m : m < \omega\}$ of eds's in ED(L) (i.e. the Gödel numbers of the S_m's form an r.e. set) such that for every L-structure $\mathfrak{A}$ and for every $a \in A^\omega$

$$\langle\mathfrak{A}, a\rangle \models \alpha \text{ iff for every } m < \omega, \langle\mathfrak{A}, a\rangle \models S_m\downarrow.$$

Proof: (1) follows from Proposition 2.2.1. (3) follows from (1) by using conjunctive normal form for open formulas. (2) follows from (3); i.e.

$S\downarrow \to Q\downarrow$ can easily be expressed as $S_* \equiv Q_*$ for some (effectively found) eds's S_*, Q_*. Finally (4) follows from (3); i.e. $S\uparrow \vee Q\downarrow$ can be easily expressed as $\bigwedge_{m<\omega} P_m\downarrow$ for an r.e. set $\{P_m : m < \omega\}$ of eds's. ∎

It is perhaps interesting to note that in general the ω in (2) or in (3) cannot be bounded by any integer. This follows from the following result due to J. Bergstra.

3.3.2 *Theorem* ([3])

There exists a finite language $\mathbb{L}$ and an $\mathbb{L}$-structure $\mathfrak{A}$ such that for every $n < \omega$ there exists an open formula $\alpha \in$ LED(L) such that for arbitrary eds's $S_0, \ldots, S_{n-1}, Q_0, \ldots, Q_{n-1}$ in ED(L),

$$\mathfrak{A} \models \alpha \leftrightarrow \bigwedge_{i<n} (S_i\downarrow \to Q_i\downarrow) \text{ } \textit{does not} \text{ hold.}$$

The proof of the above result uses a structure in which eds's define partially computable functionals on Cantor space and exploits their continuity properties. ∎

The next result is proved in the same way as the analogous result (i.e. the *prenex normal form*) for first order logic.

3.3.3 *Theorem*

Let $\mathbb{L}$ be a finite language. For every $\alpha \in$ LED(L) there exists $\alpha_* \in$ LED(L) such that

(i) $\models \alpha \leftrightarrow \alpha_*$

(ii) α_* is of the form $Q_0 x_{i_0} \ldots Q_{n-1} x_{i_{n-1}} \varphi$, where each Q_i is either $\forall$ or $\exists$, and φ is an open LED(L) formula.

3.4 *Structures Uniquely Definable in LED*

It follows from 3.2.5 that every finite structure is uniquely definable (up to isomorphism) by a single LED formula (corresponding to a first order formula). Here we investigate the following problem: what structures are uniquely axiomatizable by a single LED formula? We give a complete characterization of structures which are uniquely axiomatizable by open LED(L) formulas, in the case that $\mathbb{L}$ contains at least one constant.

Let us start with the following example.

3.4.1 *Example*

Let $L_C = \{0\}$, $L_F = \{S\}$, $\rho_L(S) = 1$, $L_R = \phi$. Define an eds $Q \in ED(L, 1)$ by $Q(n) = \langle S^n(0) = x_0, x_0 \rangle$ for all $n < \omega$.

It is easy to see that if α is:

$$Q{\downarrow} \wedge (S(x_0) \doteq S(x_1) \rightarrow x_0 \doteq x_1) \wedge (\neg S(x_0) \doteq 0)$$

then for every L-structure $\mathfrak{A}$, $\mathfrak{A} \models \alpha$ iff $\mathfrak{A} \cong \langle \omega, S, 0 \rangle$, where S is interpreted as successor.

An L-structure $\mathfrak{A}$ is said to be ***uniquely definable*** by a set Σ of LED(L) formulas if $\mathfrak{A}$ is the only model (up to isomorphism) of Σ. Since LED formulas express program properties (cf. 3.2 and 3.3), structures uniquely defined in LED can be viewed as those which are uniquely describable by their algorithmic properties. For example, STACK can be presented as a certain structure (cf. [29]) which can then be shown to be uniquely definable. More generally, it follows from Proposition 3.4.2 below that Abstract Data Types (cf. [13]) can be viewed as structures uniquely definable in LED. This also holds for the ring of integers, the field of rationals, and the field of recursive reals (cf. [1]).

3.4.2 *Proposition* ([4])

Let L be a finite language with $L_C \neq \emptyset$. Let $\mathfrak{A}$ be an L-structure. The following conditions are equivalent:

(1) $\mathfrak{A}$ is uniquely definable by a set of open LED(L) formulas.

(2) $\mathfrak{A}$ is uniquely definable by a set $\{S_i{\downarrow} : i \in I\}$ of termination predicates, where each $S_i \in ED(L)$.

(3) $\mathfrak{A}$ has no proper substructures, i.e. for every $a \in A$ there exists $t \in T(L, 0)$ with $a = t^{\mathfrak{A}}$.

Proof: (1) → (3) and (2) → (1) are obvious. For (3) → (2), we show that the property of having no proper substructures can be expressed by one formula of the form $S{\downarrow}$ and then we add formulas describing the *diagram* (see, e.g., [7] for a definition) of $\mathfrak{A}$. ∎

For an arbitrary finite language L the following analogue of 3.4.2 can be proved.

3.4.3 *Proposition* ([4])

For an arbitrary finite language L, if an L-structure $\mathfrak{A}$ is uniquely definable by a set of open LED formulas then $\mathfrak{A}$ has no proper substructures, i.e. for every substructure $\mathfrak{A}_0$ of $\mathfrak{A}$ with $\mathfrak{A}_0 \neq \emptyset$, $\mathfrak{A}_0 = \mathfrak{A}$ holds.

Proof: Let $\mathfrak{A}$ be uniquely definable by a set Σ of open LED(L) formulas. Each substructure $\mathfrak{A}_0$ of $\mathfrak{A}$ satisfies Σ as well, so $\mathfrak{A}_0 \cong \mathfrak{A}$. Since $\mathfrak{A}$ has finitely generated substructures (which must be isomorphic to $\mathfrak{A}$), we conclude that $\mathfrak{A}$ itself is finitely generated. If $\mathfrak{A}_0$ is a proper substructure of $\mathfrak{A}$ then we construct a sequence $\mathfrak{A}_0, \mathfrak{A}_1, \ldots$ of L-structures such that

(i) $\mathfrak{A}_1 = \mathfrak{A}$;

(ii) for every $n < \omega$, $\mathfrak{A}_n$ is a proper substructure of $\mathfrak{A}_{n+1}$;

(iii) for every $n < \omega$, there is an isomorphism $f_n : \mathfrak{A}_{n+1} \to \mathfrak{A}_{n+2}$ such that $f_n(\mathfrak{A}_n) = \mathfrak{A}_{n+1}$

Then $\mathfrak{A}^* = \bigcup_{n<\omega} \mathfrak{A}_n$ is not finitely generated and thus not isomorphic to $\mathfrak{A}$. On the other hand, both $\mathfrak{A}^*$ and $\mathfrak{A}$ do satisfy the same open LED(L) formulas. ∎

In order to give a characterization of structures uniquely definable by a single LED formula we introduce some standard definitions.

Let L be a finite language with $L_C \neq \emptyset$ and let $L_R = \{r_0, \ldots, r_{k-1}\}$ with $\rho_L(r_i) = n_i$ for $i < n$. Let $\mathfrak{A}$ be an L-structure without proper substructures. Let $g : \omega \to T(L, 0)$ be the recursive coding fixed in 1.5. Define a *code* for $\mathfrak{A}$ to be a k+1 tuple $C(\mathfrak{A}) = \langle C(r_0, \mathfrak{A}), \ldots, C(r_{k-1}, \mathfrak{A}), C(\approx, \mathfrak{A})\rangle$ of relations in ω, where $C(r_i, \mathfrak{A}) \subseteq \omega^{n_i}$ for $i < k$ and $C(\approx, \mathfrak{A}) \subseteq \omega^2$, satisfying the following conditions:

(i) for $i < k$ and $\langle m_0, \ldots, m_{n_i-1}\rangle \in \omega^{n_i}$,

$\langle m_0, \ldots, m_{n_i-1}\rangle \in C(r_i, \mathfrak{A})$ iff $\langle g(m_0), \ldots, g(m_{n_i-1})\rangle \in r_i^{\mathfrak{A}}$

(ii) for $\langle m_0, m_1\rangle \in \omega^2$,

$\langle m_0, m_1\rangle \in C(\approx, \mathfrak{A})$ iff $\mathfrak{A} \models g(m_0) = g(m_1)$.

For a set X, let $\mathscr{P}(X)$ denote the set of all subsets of X. Let $\mathscr{C} \subseteq \mathscr{P}(\omega^{n_0}) \times \ldots \times \mathscr{P}(\omega^{n_{k-1}}) \times \mathscr{P}(\omega^2)$ be the set of those k+1-tuples $\langle C_0, \ldots, C_{k-1}, E\rangle$ of relations in ω such that the k+1-th component E is a coding (in the sense of (ii) above) of an equivalence relation in T(L, 0) (the structure T(L, 0) has only operations and constants determined by $L_F \cup L_C$ in a natural way), and for every $i < k$ the following holds for all m, $p \in \omega^{n_i}$, if $(m_j, p_j) \in E$ for every $j < n_i$ then $m \in C_i$ iff $p \in C_i$.

Let $\mathscr{K}$ denote the class of all L-structures without proper substructures. It is easy to observe that to each $\mathscr{K}_0 \subseteq \mathscr{K}$ there corresponds in a natural way $(\mathscr{K}_0)^C = \{C(\mathfrak{A}) : \mathfrak{A} \in \mathscr{K}_0\} \subseteq \mathscr{C}$. And conversely, for every $\mathscr{C}_0 \subseteq \mathscr{C}$ there exists $\mathscr{K}_0 \subseteq \mathscr{K}$ with $(\mathscr{K}_0)^C = \mathscr{C}_0$.

A subset $X \subseteq \mathscr{P}(\omega^{n_0}) \times \ldots \times \mathscr{P}(\omega^{n_{k-1}}) \times \mathscr{P}(\omega^2)$ is said to be Π^0_2 if there exists a recursive (i.e. Δ^0_0) formula $\varphi(r_0, \ldots, r_{k-1}, \approx, x_0, x)$ such that $\langle C_0, \ldots, C_{k-1}, E\rangle \in X$ iff $\forall x_0 \exists x_1 \varphi[C_0, \ldots, C_{k-1}, E, x_0, x_1]$ is true in the standard model of arithmetic.

A subclass $\mathscr{K}_0 \subseteq \mathscr{K}$ is Π^0_2-*definable* if $(\mathscr{K}_0)^C$ is a Π^0_2 set. Observe that $\mathscr{C}$ defined above is a Π^0_2 set since the conditions defining it are Π^0_2. Therefore $\mathscr{K}$ itself is Π^0_2-definable. Finally an L-structure $\mathfrak{A} \in \mathscr{K}$ is Π^0_2-definable if $\{\mathfrak{A}\}$ is Π^0_2-definable.

3.4.4 *Theorem* ([4])

Let L be a finite language with $L_C \neq \emptyset$ and let $\mathfrak{A}$ be an L-structure. The following conditions are equivalent:

(i) $\mathfrak{A}$ is uniquely definable by a single formula of the form S+, for a certain $S \in ED(L)$

(ii) $\mathfrak{A}$ is uniquely definable by a recursively enumerable set $\{\alpha_i : i < \omega\}$ of open LED(L) formulas

(iii) $\mathfrak{A}$ has no proper substructures and $\mathfrak{A}$ is Π^0_2-definable.

Proof: (i) → (ii) is obvious. For (ii) → (iii) we apply Theorem 3.3.1(4) and transform $\{\alpha_i : i < \omega\}$ into the semantically equivalent r.e. set

$\{S_i\downarrow: i < \omega\}$ with $S_i \in ED(L)$ for $i < \omega$. Then from $\bigwedge_{K\omega} S_i\downarrow$ we get a Π_2^0 condition defining $\mathfrak{A}$. To prove (iii) → (i) we essentially reverse the proof of (ii) → (iii), applying a recursive enumeration of the finite sequences of positive integers.

In the next two sections, we exploit the observations we made in 3.2.9 about the relation of LED to $L_{\omega_1\omega}$. We apply some methods from $L_{\omega_1\omega}$ to derive results concerning LED.

3.5 *Completeness Theorem*

The purpose of this section is to present a formal proof system for LED and to prove its completeness. The idea of the proof is borrowed from the Model Theory of $L_{\omega_1\omega}$ (cf. [16]) and is based on a (modified) notion of the *consistency property*.

3.5.1 We first need a transformation for "moving the negation inside". Let $\alpha \in LED(L)$ (the language L is fixed throughout the section). Then the formula $\alpha' \in LED(L)$ is defined inductively:

$(S \doteq Q)'$ is $\neg(S \doteq Q)$,
$(\neg\alpha)'$ is α,
$(\alpha \vee \beta)'$ is $\alpha' \wedge \beta'$,
$(\alpha \wedge \beta)'$ is $\alpha' \vee \beta'$,
$(\exists x_n\alpha)'$ is $\forall x_n(\alpha')$,
$(\forall x_n\alpha)'$ is $\exists x_n(\alpha')$.

Now we are in a position to present the system.

3.5.2 *Axioms*

We have the following axiom schemes, where $\alpha \in LED(L)$ and x and y are any individual variables.

A.1 Every tautology of finitary propositional logic.

A.2 $\neg\alpha \leftrightarrow \alpha'$.

A.3 $S^{(n)} \doteq Q^{(n)} \to S \doteq Q$,
where $n < \omega$, and S, Q $\in$ ED(L).

A.4 $\forall x\alpha \rightarrow \alpha(x/t)$,
where $t \in T(L)$, t is *free* for x in α, and $\alpha(x/t)$ is obtained by replacing each free occurrence of x in α by t.

A.5 $x \doteq x$.

A.6 $x \doteq y \rightarrow y \doteq x$.

A.7 $(\alpha \wedge t \doteq x) \rightarrow \alpha(x/t)$.

3.5.3 *Rules of Inference*

In the rules below, α, $\beta \in$ LED(L) and x is any individual variable.

R.1 $$\frac{\alpha,\ \alpha\rightarrow\beta}{\beta}$$

R.2 $$\frac{\alpha \rightarrow \beta}{\alpha\rightarrow\forall x\beta}$$

where x does not occur free in α.

R.3 $$\frac{\{\alpha \rightarrow \neg(S^n \doteq Q^n) : n < \omega\}}{\alpha \rightarrow \neg(S \doteq Q)}$$

where S, Q $\in$ ED(L).

Let $\Sigma \subseteq$ LED(L) and let $\alpha \in$ LED(L). Then α is said to be *provable from* Σ, in symbols $\Sigma \vdash_{LED(L)} \alpha$, if α belongs to the least set of LED(L) formulas which contains Σ, all the axioms obtained from schemes A.1 - A.7, and which is closed under the rules of inference R.1 - R.3. We write $\Sigma \models \alpha$ if every model for Σ is a model for α.

3.5.4 *LED Over Arbitrary Languages*

Our proof of the completeness of the above formal system will require us to work with countable languages rather than with finite ones, so we define here LED(L) for an arbitrary language L to be $\bigcup\{LED(L_0) : L_0 \subseteq L$ and L_0 is finite$\}$. Observe that all the notions introduced in section 3.5 make sense for arbitrary languages, in particular the notion of provability in the formal system (3.5.2, 3.5.3).

The result below has been proved (in [32]) for a three-valued logic of programs, but only for finite sets Σ. It has also been relativized (in [33]) to LED, but again only for finite sets Σ.

3.5.5 *Theorem*

Let L be a countable language. For every $\Sigma \subseteq LED(L)$ and for every $\alpha \in LED(L)$, $\Sigma \vdash_{LED(L)} \alpha$ iff $\Sigma \models \alpha$.

To prove 3.5.5 we first provide a tool for constructing models of LED formulas, the *Model Existence Theorem*, an analogous result to that for $L_{\omega_1\omega}$. This theorem is based on the notion of a *consistency property*. The reader may compare the consistency property for LED with the corresponding property for $L_{\omega_1\omega}$ (cf. [16]).

3.5.6 *Consistency Property*

Let L be a countable language. Let L^* denote the language obtained from L by adding a countable set C of constant symbols. Let U be a set of countable subsets of $LED(L^*)$. U is said to have the *consistency property* iff for each $u \in U$ and for arbitrary $\alpha, \beta \in LED(L^*)$, all the following hold.

C.1 (Consistency Rule) Either $\alpha \notin u$ or $\neg\alpha \notin u$.

C.2 (′ - Rule) If $\neg\alpha \in u$ then $u \cup \{\alpha'\} \in U$.

C.3 ($\wedge$ - Rule) If $(\alpha \wedge \beta) \in u$ then $u \cup \{\alpha\} \in U$ and $u \cup \{\beta\} \in U$.

C.4 ($\forall$ - Rule) If $(\forall x_n \alpha) \in u$ then for all $c \in C$, $u \cup \{\alpha(x_n/c)\} \in U$.

C.5 ($\vee$ - Rule) If $(\alpha \vee \beta) \in u$ then either $u \cup \{\alpha\} \in U$ or $u \cup \{\beta\} \in U$.

C.6 ($\exists$ - Rule) If $(\exists x_n \alpha) \in u$ then for some $c \in C$ $u \cup \{\alpha(x_n/c)\} \in U$.

C.7 (Convergence Rule) For $S, Q \in ED(L^*)$, if $(S \doteq Q) \in u$ then for some $n < \omega$, $u \cup \{S^{(n)} \doteq Q^{(n)}\} \in U$.

C.8 (Divergence Rule) For $S, Q \in ED(L^*)$, if $\neg(S \doteq Q) \in u$ then for all n, ω, $u \cup \{\neg(S^{(n)} \doteq Q^{(n)})\} \in U$.

By a *basic term* we mean either a constant symbol or a term of the form $f(c_1, \ldots, c_n)$ where $f \in L_F$, $\rho(f) = n$, and $c_1, \ldots, c_n \in C$.

C.9 (Equality Rules) Let t be a basic term, and $c, d \in C$, $u \in U$
If $(c \doteq d) \in u$ then $u \cup \{d \doteq c\} \in U$.
If $c \doteq t$, $\alpha(x/t) \in u$ then $u \cup \{\alpha(x/c)\} \in U$.
For some $e \in C$, $u \cup \{e \doteq t\} \in U$.

3.5.7 ***Model Existence Theorem***

If U has the consistency property and $u \in U$, then u has a model.

Proof: The proof is essentially the same as that of the analogous result for $L_{\omega_1\omega}$ (cf. [16]). ∎

3.5.8 ***Proposition***

If $\Gamma \subseteq LED(L)$ and $\beta \in LED(L)$, then $\Gamma \vdash_{LED(L^*)} \beta$ iff $\Gamma \vdash_{LED(L)} \beta$.

Proof: Suppose $\Gamma \vdash \beta$ in $LED(L^*)$. Since proofs in LED are particular instances of proofs in $L_{\omega_1\omega}$, it follows from *cut elimination* for $L_{\omega_1\omega}$ (cf. [20]) that there is a proof of β from Γ in $LED(L^*)$ which uses no constant symbols from C. Therefore $\Gamma \vdash \beta$ in LED(L). ∎

3.5.9 Now we are in position to prove 3.5.5. If $\Sigma \vdash \alpha$ in LED(L) then obviously $\Sigma \models \alpha$. Now suppose $\Sigma \nvdash \alpha$ in LED(L). Let Σ^* be the set of all universal closures of formulas in Σ. Obviously $\Sigma^* \nvdash \alpha$ in LED(L). Let $U = \{\Sigma^* \cup u : u$ is a finite set of sentences in $LED(L^*)$, and $\Sigma^* \nvdash_{LED(L^*)} \neg(\bigwedge_{\sigma \in u}\sigma)\}$. By 3.5.8, $\Sigma^* \cup \{\neg\alpha\} \in U$. Since L is countable, every element of U is also countable. One can easily check that U has the consistency property. Therefore, by 3.5.7, $\Sigma^* \cup \{\neg\alpha\}$ has a model. ∎

Remark: We essentially needed the cut elimination theorem in the proof of 3.5.8 only for the case where there are only finitely many individual variables

which do not occur (free or bound) in the formulas of Γ. In particular this proof does not use cut elimination when Γ is finite.

3.6 *The Hanf Number of LED*

In this section we investigate the Löwenheim-Skolem theorems in LED. Because the downward Löwenheim-Skolem theorem is true for $L_{\omega_1\omega}$ (cf. [16]), it remains true for LED (cf. 3.2.9). Thus we get

3.6.1 *Theorem*

For every finite language L and for every $\Sigma \subseteq \mathrm{LED}(L)$, if Σ has an infinite model then it has a countable model.

We have already seen in 3.4.1 that the upward Löwenheim-Skolem Theorem fails for LED. Let L be a finite language. The *Hanf number* of LED(L) (cf. [16]) is the least cardinal κ such that for each $\varphi \in \mathrm{LED}(L)$, if φ has a model of power $\geq \kappa$ then φ has arbitrary large models.

The cardinals $\beth_\alpha$, for α an ordinal, are defined inductively:

$$\beth_0 = \omega,\ \beth_{\alpha+1} = 2^{\beth_\alpha},\ \beth_\alpha = \cup_{\beta<\alpha}\beth_\beta \text{ when } \alpha \text{ is a limit ordinal.}$$

An ordinal α is said to be a *recursive ordinal* (cf. [28]) if there is a recursive binary relation $R \subseteq \omega^2$ such that R is a well-ordering of type α. Let ω_1^{CK} be the first non-recursive ordinal.

The main result of this section is

3.6.2 *Theorem* ([33])

Let L be a finite language containing at least one constant symbol, two unary function symbols, and one binary predicate symbol. Then the Hanf number of LED(L) is $\beth_{\omega_1^{CK}}$.

Proof: Let κ denote the Hanf number of LED(L). First we show that $\kappa \leq \beth_{\omega_1^{CK}}$. Let $L^{CK}_{\omega_1\omega}$ be the predicate calculus with recursively enumerable disjunctions allowed (cf. [16]). It is easy to check that LED(L) is interpretable (as in 3.2.9) in $L^{CK}_{\omega_1\omega}$. By the Morley-Barwise theorem (cf. [16], Thm. 22), which says that the Hanf number of $L^{CK}_{\omega_1\omega}$ is $\beth_{\omega_1^{CK}}$, it follows $\kappa \leq \beth_{\omega_1^{CK}}$. The inequality $\beth_{\omega_1^{CK}} \leq \kappa$ follows from the next result. ∎

3.6.3 *Proposition* ([33])

Let L be a language as in 3.6.2. Then for every recursive ordinal α there is a $\varphi \in$ LED(L) which has a model of size $\beth_\alpha$, and has no model of size $> \beth_\alpha$.

Proof: We modify the example due to Morley (cf. [16], p. 70) of a sentence φ in $L_{\omega_1\omega}$ with the required properties. Details are given in [33]. ∎

4. *Correctness Theories vs. Program Equivalence*

In this part we investigate the relationships between the notions introduced in 3.2.1, 3.2.2, 3.2.3, 3.2.7, and 3.2.8. It turns out that this investigation leads to mathematically deep and interesting questions. The author would again like to thank Albert R. Meyer, who suggested this investigation.

4.1 *General Relationships*

We first introduce some notation. Let L be a finite language. Let $\mathcal{K}$ be a class of L-structures. Recall that for $\alpha \in$ LED(L), $\mathcal{K} \models \alpha$ means that every $\mathfrak{A} \in \mathcal{K}$ is a model for α. If $\Sigma \subseteq$ LED(L), then by Mod(Σ) we denote the class of all models for Σ.

Let S $\in$ ED(L, n). By the *partial correctness theory* of S with respect to $\mathcal{K}$ we mean the set $PC(S, \mathcal{K}) = \{\langle\alpha, \beta\rangle \in L_{\omega\omega}(L, n) \times L_{\omega\omega}(L, n+1): \mathcal{K} \models \alpha \rightarrow \forall x_n(S \doteq x_n \rightarrow \beta)\}$ (this set is denoted in [6] as $MPC_{\mathcal{K}}(S)$).

By the *total correctness theory* of S with respect to $\mathcal{K}$ we mean the set $TC(S, \mathcal{K}) = \{\langle\alpha, \beta\rangle \in L_{\omega\omega}(L, n) \times L_{\omega\omega}(L, n+1) : \mathcal{K} \models \alpha \rightarrow \exists x_n(S \doteq x_n \wedge \beta)\}$ (this set is denoted in [6] as $MTC_{\mathcal{K}}(S)$).

4.1.1 *Theorem*

For an arbitrary finite language L, $n < \omega$, S, Q $\in$ ED(L, n), and a class $\mathcal{K}$ of L-structures, all of the following hold.

(i) If $\mathcal{K} \models S \doteq Q$ then $TC(S, \mathcal{K}) = TC(Q, \mathcal{K})$.
(ii) $TC(S, \mathcal{K}) = TC(Q, \mathcal{K})$ iff $\mathcal{K} \models S \equiv Q$.
(iii) If $\mathcal{K} \models S \equiv Q$ then $PC(S, \mathcal{K}) = PC(Q, \mathcal{K})$.
(iv) If $PC(S, \mathcal{K}) = PC(Q, \mathcal{K})$ then $\mathcal{K} \models S \sim Q$.

Proof: Straightforward. ∎

4.1.2 We now consider the converses of (i), (iii) and (iv) of 4.1.1. Let LOOP be an eds defined by $LOOP(m) = \langle\neg(x_0 = x_0), x_0\rangle$ for $m < \omega$. Obviously in every L-structure $\mathfrak{A}$, $LOOP^{\mathfrak{A}} = \emptyset$. Now, if $\mathcal{K} \neq \emptyset$ then the implication in (i) cannot be reversed for trivial reasons -- clearly $TC(LOOP, \mathcal{K}) = TC(LOOP, \mathcal{K})$ but $\mathcal{K} \not\models LOOP \doteq LOOP$. For (iv) it is enough to observe that for every S $\in$

ED(L, 1), $\mathcal{K} \models$ LOOP ~ S but $\langle x_0 = x_0, \neg(x_0 = x_0)\rangle \in PC(S, \mathcal{K})$ iff $\mathcal{K} \models$ LOOP $\equiv$ S. This observation gives rise to many counter-examples. The subsections 4.2 through 4.5 are devoted to investigating the question: for what classes $\mathcal{K}$ can the implication in (iii) be reversed to hold for arbitrary S, Q $\in$ ED(L)? In 4.2 we shall see that in general (iii) cannot be reversed. On the other hand, in 4.3 we see that if we allow language extensions to express partial correctness conditions then (iii) can be reversed for an arbitrary (first-order) elementary class $\mathcal{K}$. Finally, 4.4 shows that for the class of all L-structures "partial correctness determines the semantics".

A class $\mathcal{K}$ of L-structures is said to be *eds-complete* if for arbitrary S, Q $\in$ ED(L), $PC(S, \mathcal{K}) = PC(Q, \mathcal{K})$ implies $\mathcal{K} \models S \equiv Q$.

4.2 *Determinateness on Elementary Classes*

A class $\mathcal{K}$ of L-structures is said to be *elementary* if for some $\Sigma \subseteq L_{\omega\omega}(L)$, $\mathcal{K} = Mod(\Sigma)$. In this subsection we investigate the question: when is a given elementary class eds-complete? The first result shows that even very simple elementary classes need not be eds-complete.

4.2.1 *Theorem* ([6])

Let L be the following language: $L_C = L_R = \emptyset$, $L_F = \{f, g\}$, $\rho_L(f) = \rho_L(g) = 1$. Then the class $\mathcal{K} = Mod(\{\forall x_0(f(g(x_0)) = g(f(x_0)) = x_0)\})$ is not eds-complete.

Proof: Take S to compute the two argument projection function $S(x_0, x_1) = x_0$. Let Q $\in$ ED(L, 2) be an eds that does the following: given two arguments, it checks whether the first argument generates a finite subalgebra, or the second argument generates a finite subalgebra, or the first argument belongs to the subalgebra generated by second argument, or the second argument belongs to the subalgebra generated by the first argument. If any of the above conditions hold then it gives as output the first argument, otherwise it diverges.

Clearly $\mathcal{K} \not\models S \equiv Q$. To prove $PC(S, \mathcal{K}) = PC(Q, \mathcal{K})$ observe first that $PC(S, \mathcal{K}) \subseteq PC(Q, \mathcal{K})$ obviously holds. If $\langle\alpha, \beta\rangle \in PC(Q, \mathcal{K}) - PC(S, \mathcal{K})$ then we have

$\mathcal{K} \models (\alpha \wedge \neg\beta(x_2/x_0)) \rightarrow Q\uparrow$ and

$\mathcal{K} \not\models \alpha \rightarrow \beta(x_2/x_0)$.

Let $\gamma \in L_{\omega\omega}(L, 2)$ be the formula $\alpha \wedge \neg\beta(x_2/x_0)$.
By a standard application of the Compactness Theorem [7, p.67] we find a model $\mathfrak{A} \in \mathcal{K}$ and three elements $a, b, c \in \mathfrak{A}$ such that

$$(*) \quad \mathfrak{A} \models \gamma[a, b], \text{ and } \mathfrak{A} \models \neg\gamma[a, c]$$

and the subalgebras generated by a, b, c are all infinite and pairwise disjoint. Then from the specification of $\mathcal{K}$ it follows that there is an automorphism $h : \mathfrak{A} \rightarrow \mathfrak{A}$ such that $h(a) = a$ and $h(b) = c$. But this contradicts (*). ∎

The main result of this subsection is the following.

4.2.2 ***Theorem*** **([6])**

Let $\mathcal{K}$ be a nonempty elementary class of L-structures. For S, Q $\in$ ED(L), if PC(S, $\mathcal{K}$) = PC(Q, $\mathcal{K}$) then there exists a countable $\mathfrak{A} \in \mathcal{K}$ such that $\mathfrak{A} \models S \equiv Q$.

Proof: Let S, Q $\in$ ED(L). Define a countable family $\{\Gamma_n : n \in \omega\}$ of sets of open first order formulas in $L_{\omega\omega}(L)$. For $n \in \omega$ we set

$$\Gamma_{2n} = \{(\neg\alpha_{S,m} \vee \neg(t_{S,m} = t_{Q,n})) \wedge \alpha_{Q,n} : m < \omega\},$$

$$\Gamma_{2n+1} = \{(\neg\alpha_{Q,m} \vee \neg(t_{Q,m} = t_{S,n})) \wedge \alpha_{S,n} : m < \omega\}.$$

Let $\Sigma \subseteq L_{\omega\omega}(L)$ be such that $\mathcal{K} = \mathrm{Mod}(\Sigma)$.

By routine computations one can prove the following two results. (See [7] for the necessary model theoretic definitions).

Proposition A

Let $\mathcal{K}$ and Σ be as above. If S, Q $\in$ ED(L) are deterministic (cf. 2.1) then the following conditions are equivalent:

(i_A) $PC(S, \mathscr{K}) = PC(Q, \mathscr{K})$.

(ii_A) For every $n < \omega$, Σ locally omits Γ_n.

Proposition B

Let $\mathfrak{A}$ be an L-structure. If S, Q $\in$ ED(L) are deterministic then the following conditions are equivalent:

(i_B) $\mathfrak{A} \models S \equiv Q$.

(ii_B) For every $n < \omega$, $\mathfrak{A}$ omits Γ_n.

Now 4.2.2 follows from 2.1.2(i), propositions A and B, and the *Omitting Types Theorem* (cf. [7], Thm. 2.2.15). ∎

In the rest of this subsection we derive some corollaries from 4.2.2.

Call an elementary class $\mathscr{K}$ of L-structures *complete* if any two elements of $\mathscr{K}$ satisfy exactly the same sentences in $L_{\omega\omega}(L)$.

4.2.3 *Corollary* ([6])

Let $\mathscr{K}$ be a nonempty complete elementary class of L-structures. The following conditions are equivalent for arbitrary S, Q $\in$ ED(L).

(i) $PC(S, \mathscr{K}) = PC(Q, \mathscr{K})$.

(ii) For some countable $\mathfrak{A} \in \mathscr{K}$, $\mathfrak{A} \models S \equiv Q$.

(iii) For some countable $\mathfrak{A} \in \mathscr{K}$, $PC(S, \{\mathfrak{A}\}) = PC(Q, \{\mathfrak{A}\})$.

Proof: By 4.2.2 and 4.1.1, (i) → (ii) and (ii) → (iii) hold (we have not yet used the assumption that $\mathscr{K}$ is complete). Implication (iii) → (i) follows from the following easy fact. If $\mathscr{K}$ is a complete elementary class of L-structures then for every S $\in$ ED(L) and for every $\mathfrak{A} \in \mathscr{K}$, $PC(S, \mathscr{K}) = PC(S, \{\mathfrak{A}\})$. ∎

4.2.4 *Corollary* ([6])

Let $\mathcal{K}$ be an ω-categorical complete elementary class of L-structures (i.e. $\mathcal{K}$ contains a countable L-structure and any two countable elements of $\mathcal{K}$ are isomorphic). Then $\mathcal{K}$ is eds-complete.

Proof: Follows immediately from 4.2.3 and 3.6.1. ∎

Call a class $\mathcal{K}$ of L-structure Π_1^{LED}-complete if all elements of $\mathcal{K}$ have the same termination properties, i.e. for every $S \in ED(L)$ and for arbitrary $\mathfrak{A}_1, \mathfrak{A}_2 \in \mathcal{K}$, $\mathfrak{A}_1 \models S\!\downarrow$ iff $\mathfrak{A}_2 \models S\!\downarrow$.

4.2.5 *Corollary*

Let $\mathcal{K}$ be a nonempty elementary class of L-structures. If $\mathcal{K}$ is Π_1^{LED}-complete then $\mathcal{K}$ is eds-complete.

Proof: Follows from 4.2.2 and 3.3.1(4). ∎

An L-structure $\mathfrak{A}$ is said to be ***algorithmically trivial*** if for every $S \in ED(L)$, if $\mathfrak{A} \models S\!\downarrow$ then for some $n < \omega$, $\mathfrak{A} \models S \equiv S^{(n)}$. Algorithmically trivial structures have been investigated by many authors (cf. [10, 17, 18, 19, 34]). For a survey of results, including some new ones, the reader should consult [34].

The next result gives a full characterization of the eds-complete classes among all elementary complete ones.

4.2.6 *Theorem*

For a complete elementary class $\mathcal{K}$ all the following conditions are equivalent:

(i) $\mathcal{K}$ is eds-complete.

(ii) $\mathcal{K}$ is Π_1^{LED}-complete.

(iii) Every $\mathfrak{A} \in \mathcal{K}$ is algorithmically trivial.

Proof: The only interesting case is when $\mathcal{K}$ is a class of infinite structures.

(i) → (iii). If $\mathfrak{A} \in \mathcal{K}$ is not algorithmically trivial then there exists m with $0 < m < \omega$, and $S \in ED(L, m)$ such that $\mathfrak{A} \models S\downarrow$ but for $n < \omega$, $\mathfrak{A} \not\models S^{(n)}\downarrow$. Without loss of generality, S can be chosen so that it computes a partial projection on the first component. Let $Q(x_0, \ldots, x_{m-1}) = x_0$ be a total eds which computes this projection. Since S is not equivalent to any of its finite parts in $\mathfrak{A}$, by a standard compactness argument there is $\mathfrak{B} \in \mathcal{K}$ such that S is not total on $\mathfrak{B}$. Here we get a contradiction since $\mathfrak{A} \models S \equiv Q$, so by 3.6.1 $\mathfrak{A}$ can be assumed to be countable. Then by 4.1.1(iii) $PC(S, \{\mathfrak{A}\}) = PC(Q, \{\mathfrak{A}\})$, and by 4.2.3 $PC(S, \mathcal{K}) = PC(Q, \mathcal{K})$, so $\mathcal{K} \models S \equiv Q$ by eds-completeness and $\mathfrak{B} \models S \equiv Q$, a contradiction.

(iii) → (ii) because $\mathcal{K}$ is complete.

(ii) → (i) follows from 4.2.5. ∎

Below we give two examples which are immediately derivable from 4.2.6. For an L-structure $\mathfrak{A}$, $Th(\mathfrak{A})$ is $\{\varphi \in L_{\omega_1\omega}(L): \mathfrak{A} \models \varphi\}$.

4.2.7 Let $\mathbb{C}$ be the field of complex numbers. Then $\mathcal{K} = Mod(Th(\mathbb{C}))$ is not eds-complete (it follows from [18] that $\mathcal{K}$ contains a field which is not algorithmically trivial).

4.2.8 Let $\mathfrak{A}$ be an L-structure without proper substructures. Then $Mod(Th(\mathfrak{A}))$ is not eds-complete (by 3.4.2 $\mathfrak{A}$ is not algorithmically trivial).

4.3 *Determinateness via Language Extensions*

Let L be a finite language. Let N be the language of arithmetic, i.e. $N_P = \{S, +, \cdot\}$, $\rho_N(S) = 1$, $\rho_N(+) = \rho_N(\cdot) = 2$, $N_C = \{0\}$. Assume that L and N are disjoint and let L(N) be the extension of L by N. If $\mathcal{K}$ is a class of L-structures let $\mathcal{K}(N)$ be the class of all L(N) expansions of structures in $\mathcal{K}$.

The aim of this subsection is to sketch a proof of the following result, which has been obtained independently and at about the same time in [6] and [22].

4.3.1 *Theorem* ([6, 22])

Let $\mathscr{K}$ be an arbitrary class of countable L-structures. Then for arbitrary S, Q $\in$ ED(L), if $PC(S, \mathscr{K}(N)) = PC(Q, \mathscr{K}(N))$ then $\mathscr{K} \models S \equiv Q$.

Proof: A moment of reflection upon 4.1.1(iv) shows that in order to prove 4.3.1 it is sufficient (and necessary) to prove the following result.

4.3.2 *Theorem*

Let $\mathscr{K}$ be an arbitrary class of countable L-structures. Then if $S \in ED(L, n)$ for some $n < \omega$, and if for some $\mathfrak{A} \in \mathscr{K}$ and for some $a \in A^n$, $\langle\mathfrak{A}, a\rangle \models S^{\dagger}$ then there exists a formula $\varphi \in L_{\omega\omega}(L(N), n)$ such that

(i) $\mathscr{K}(N) \models \varphi \rightarrow S^{\dagger}$.

(ii) φ is consistent with $\mathscr{K}(N)$; i.e. for some $\mathfrak{B} \in \mathscr{K}(N)$, $\mathfrak{B} \models \exists x_0 \ldots \exists x_{n-1}\varphi$.

First we introduce some terminology. An L(N)-structure $\mathfrak{A}$ is said to be *standard* if its reduct to an N-structure is the standard model of arithmetic (i.e. $\mathfrak{A}|_N = \langle\omega, 0, S, +, \cdot\rangle$). For $m < \omega$, $\mathbf{m}$ is an abbreviation for the term $S^m(0)$. The next result is the key lemma for the proof of 4.3.2.

4.3.3 *Lemma* ([6, 22])

Let $\{\alpha_m : m < \omega\} \subseteq OF(L, n)$ be a recursively enumerable set of formulas. Then there exists a sentence $\psi \in L_{\omega\omega}(L(N), 0)$ and a formula $\gamma \in L_{\omega\omega}(L(N), n+1)$ such that

(i) ψ is true in all standard L(N)-structures.

(ii) For each $m < \omega$, $\models \psi \rightarrow (\alpha_m \leftrightarrow \gamma(x_n/\mathbf{m}))$.

Proof: The proof of this lemma is an adaptation of the usual proof of the representation of recursive functions in arithmetic. Details are omitted. ∎

Proof of 4.3.2: Let $S \in ED(L, n)$. Apply 4.3.3 to the r.e. set $\{\neg\alpha_{S,m} : m < \omega\}$. Let ψ and γ be formulas obtained from 4.3.3. Then $\psi \wedge \forall x_n\gamma$ satisfies (i) and (ii) of 4.3.2. ∎

To relate 4.3.1 to arbitrary elementary classes we state the following auxiliary result.

4.3.4 *Proposition*

Let $\mathcal{K}$ be an elementary class of L-structures which contains a countable structure. Let $\mathcal{K}_0$ be the class of all countable structures in $\mathcal{K}$. Then for an arbitrary $S \in ED(L)$, $PC(S, \mathcal{K}) = PC(S, \mathcal{K}_0)$.

Proof: Follows from 3.6.1. ∎

4.3.5 *Corollary* ([6], [22])

Let $\mathcal{K}$ be an elementary class of L-structures. Then for arbitrary $S, Q \in ED(L)$, if $PC(S, \mathcal{K}(N)) = PC(Q, \mathcal{K}(N))$ then $\mathcal{K} \models S \equiv Q$.

One may ask similar questions to those in 4.1, 4.2, 4.3 for definitional schemes which need not to be effective, i.e. in the definition in 2.1 the function S is arbitrary. It turns out that all results of 4.1 and 4.2 carry over to this more general notion with unchanged proofs. However our method of proving 4.3.5 essentially depends on effectiveness of a given scheme. The next result shows that 4.3.5 is no longer true for arbitrary definitional schemes, even for an elementary class which is eds-complete (cf. subsection 4.4). Let I be the finite scheme in $ED(L, 1)$ which computes the identity, i.e. $I(n) = \langle x_0 = x_0, x_0 \rangle$ for $n < \omega$. For a finite language L, let Struct(L) be the class of all L-structures.

4.3.6 *Theorem*

Let L be a finite language with $L_F = \{f_0, \ldots, f_{n-1}\}$, $L_R = \{r_0, \ldots, r_{m-1}\}$ for some $m, n < \omega$. Let L satisfy one of the following conditions

(i) $\Sigma_{i<n} \rho_L(f_i) \geq 2$, or

(ii) $\Sigma_{i<n} \rho_L(f_i) \geq 1$ and $\Sigma_{i<m} \rho_L(r_i) \geq 1$.

Then there exists a (noneffective) definitional scheme S over L, such that for an arbitrary extension $L \subseteq L^*$, the following two conditions hold:

(*) $PC(S, \mathrm{Struct}(L^*)) = PC(I, \mathrm{Struct}(L^*))$.

(**) For some $\mathfrak{A} \in \mathrm{Struct}(L)$, $\mathfrak{A} \models \exists x_0 S\uparrow$.

Proof: We sketch here a proof of 4.3.6 for the case where L contains two unary function symbols f and g. The proof for other cases mentioned in (i), (ii) is essentially the same. Let S be a scheme defined as follows, $S(n) = \langle \alpha_n, x_0 \rangle$ for $n < \omega$, where α_0 is $x_0 = f(x_0)$ and for $n \in \omega$,

$$\alpha_{n+1} = \begin{cases} \neg(g(f^n(x_0)) = x_0), & \text{if } \varphi_n(n)\downarrow, \\ \neg(g(f^n(x_0)) = f(x_0)), & \text{if } \varphi_n(n)\uparrow, \end{cases}$$

where φ_n is the n-th partial recursive function in a standard enumeration.

To prove (*) let us take any extension $L \subseteq L^*$. If $\langle \alpha, \beta \rangle \in PC(S, \mathrm{Struct}(L^*)) - PC(I, \mathrm{Struct}(L^*))$ then

(4.3.7) $\models (\alpha \wedge \neg\beta(x_1/x_0)) \rightarrow S\uparrow$

and

(4.3.8) for some $\mathfrak{A} \in \mathrm{Struct}(L^*)$, and for some $a \in A$,

$$\mathfrak{A} \models \alpha[a] \wedge \neg\beta[a, a].$$

Extend L^* by a new constant symbol c. Let γ be the sentence $\alpha(c) \wedge \neg\beta(c, c)$. Thus γ is consistent and

(4.3.9) $\models \gamma \rightarrow \neg\alpha_n(c)$ for every $n < \omega$

Let $\mathrm{Thm}(\gamma)$ be the set of all theorems deducible from γ. By the completeness theorem, $\neg\alpha_n(c) \in \mathrm{Thm}(\gamma)$ for every $n < \omega$. Let $\Gamma = \{g(f^n(c)) = c : n < \omega\} \cup \{g(f^n(c)) = f(c) : n < \omega\} \cup \{\neg f(c) = c\}$. Obviously $\Gamma \cap \mathrm{Thm}(\gamma)$ is a recursively enumerable set. Moreover $\{\neg\alpha_n(c) : n < \omega\} \subseteq \Gamma \cap \mathrm{Thm}(\gamma)$, and the reverse inclusion holds as well, by the consistency of γ. This gives us a contradiction. Condition (**) is obvious. ∎

4.4 *Struct(L) is eds-complete*

In this subsection we briefly sketch a proof of the following result (cf. [6]; a similar result appears in [22]).

4.4.1 *Theorem*

Let L be an arbitrary finite language *other than* the one where $L_F = \{f\}$, $\rho_L(f) = 1$, $L_R \neq \phi$, and for all $r \in L_R$, $\rho_L(r) = 1$. Then Struct(L) is eds-complete.

Proof: The proof is quite technically involved. Here we only sketch the main ideas behind the proof -- the details can be found in [6]. In [6] there is a complete proof of 4.4.1 for the languages L with $L_R = \phi$ (i.e. for algebraic signatures), and the methods applied in [6] can be easily adapted to all languages except the case mentioned in the hypothesis of 4.4.1.

The proof is essentially divided into two parts according to the cases the language L satisfies.

(I) $L_R = \phi$, $L_F = \{f\}$, $\rho_L(f) = 1$.

(II) {Remaining cases} - {Exceptional case}.

Actually, for case (I) a stronger result can be proved. (Note that Theorem 4.2.1 shows that the result below fails for L containing two unary function symbols).

4.4.2 *Theorem* ([6])

Let L be a language satisfying (I). Then every class $\mathcal{K}$ of L-structures which is closed under substructures is eds-complete.

Proof: There are no tricks involved, but some work needs to be done. See [6] for details. ∎

Proof of 4.4.1 (continued)

The main difficulties are found in case (II). In this case we proceed as follows.

We first prove an auxiliary result (the Localization Lemma in [6]) which makes it possible to restrict our attention to closed eds's.

Localization Lemma

Let L be a finite language and let $\mathcal{K}$ be an arbitrary class of L-structures. The following conditions are equivalent

(i) $\mathcal{K}$ is eds-complete.

(ii) For all finite extensions of L by constants to L^c and for each $S \in ED(L^c, 0)$, if $\mathfrak{A} \models S^+$ for some $\mathfrak{A} \in \mathcal{K}(L^c)$, then there is a sentence $\varphi \in L_{\omega\omega}(L^c, 0)$ which has a model in $\mathcal{K}(L^c)$ and $\mathcal{K}(L^c) \models \varphi \rightarrow S^+$ (where $\mathcal{K}(L^c)$ is the class of all expansions of structures in $\mathcal{K}$ to L^c-structures).

Proof: See [6] for details. ∎

Let L be a language satisfying (II), and let L^c be any finite extension of L by constants. We expand L^c to a *two-sorted language* L^{c^*} by adding to L^c a new sort called *SETS*, and renaming as *DOM* the sort of L. We also add $\in$, a binary relation on *SETS* × *SETS* and MAP, a binary relation on *SETS* × *DOM*. $L_{\omega\omega}(L^{c^*})$ is defined as usual; it has two types of variables, one type ranging over *DOM* and one over *SETS*.

Every four-tuple of formulas $t = \langle \alpha_D, \alpha_S, \alpha_\epsilon, \alpha_M \rangle$ in $(L_{\omega\omega}(L^c, 1))^2 \times (L_{\omega\omega}(L^c, 2))^2$ determines an interpretation H^t of $L_{\omega\omega}(L^{c^*})$ into $L_{\omega\omega}(L^c)$ in an obvious way.

The key lemma for the proof of 4.4.1 is the following:

Lemma A

Let $\gamma \in L_{\omega\omega}(L^c, 0)$ and let H^t be an interpretation of $L_{\omega\omega}(L^{c^*})$ into $L_{\omega\omega}(L^c)$. Let γ and H^t satisfy these two conditions:

(*) for every $S \in ED(L^c, 0)$ if there is an L^c-structure $\mathfrak{A}$ such that $\mathfrak{A} \models S^+$ then there exists an L^c-structure $\mathfrak{B}$ with $\mathfrak{B} \models \gamma \wedge S^+$.

(**) for every sentence $\psi \in L_{\omega\omega}(L^{c^*}, 0)$, if $\gamma \wedge \psi$ has a model (in Struct(L^{c^*})) then $H^t(\gamma \wedge \psi)$ has a model (in Struct(L^c)).

Then given any eds $S \in ED(L^c, 0)$ which diverges on a certain L^c-structure, there exists a sentence $\varphi \in L_{\omega\omega}(L^c, 0)$ which has a model in Struct(L^c) such that Struct(L^c) $\models \varphi \to S\uparrow$.

Now, the proof of 4.4.1 will follow from Lemma A and the Localization Lemma once we show that for each language L^c we can find a sentence γ and an interpretation H^t which satisfy conditions (*) and (**) (see [6] for details).

For the proof of Lemma A we first extend L^c by the language of arithmetic N. Let $S \in ED(L^c, 0)$ be an eds which diverges on a certain L^c-structure. Then we apply Theorem 4.3.2 to get a sentence $\varphi' \in L_{\omega\omega}(L^c(N), 0)$ which is consistent with Struct($L^c(N)$) and such that Struct($L^c(N)$) $\models \varphi' \to S\uparrow$.

To eliminate the symbols of N in φ' we use γ and H^t from the hypothesis of Lemma A and apply the following technical result in axiomatic set theory.

Lemma B

Let L be a finite language and let $\rho \in L_{\omega\omega}(L, 0)$.
Then there exist a sentence p of the first order language of Zermelo-Fraenkel set theory, L(ZF) and a formula $p'(x)$ of L(ZF) such that

(i) $ZF \vdash p$.

(ii) If $\mathfrak{B} \models p$ then for $b \in \mathfrak{B}$, $\mathfrak{B} \models p'[b]$ iff b is isomorphic to an L-structure which satisfies the sentence ρ. ∎

This completes the sketch of the proof of 4.4.1. The method described above does not work in the exceptional case mentioned in 4.4.1. We leave as an open problem (cf. Section 5) the question of eds-completeness of Struct(L) for languages L which are not covered by 4.4.1.

4.5 *Arithmetic Programs*

In this subsection we provide a partial result which is motivated by the following question: *is the class of all models $\mathcal{PA}$ of Peano Arithmetic eds-complete?*

Let L be a language with $L_C = \{0\}$, $L_F = \{s_0 +, \cdot\}$, $L_R = \{\leq\}$, $\rho_L(S) = 1$, $\rho_L(+) = \rho_L(\leq) = 2$. Let $\mathfrak{N} = \langle \omega, 0, S, +, \cdot, \leq \rangle$ be the standard model of arithmetic. Let $\mathcal{PA}$ be the class of all models of Peano arithmetic. Let $\mathcal{K} \subseteq \mathcal{PA}$ be the class of all L-structures which have the same first order universal sentences in $L_{\omega\omega}(L)$ true as the standard model $\mathfrak{N}$.

For the proof of the next result, Gödel's Incompleteness Theorem is used. We refer the reader to [6] for details.

4.5.1 ***Theorem*** **([6])**

For arbitrary S, Q $\in$ ED(L), PC(S, $\mathcal{PA}$) = PC(Q, $\mathcal{PA}$) implies $\mathcal{K} \models S \equiv Q$.

5. *Open Problems*

Below we list some problems which are naturally connected with topics we have discussed in the paper.

5.1 *Fragments of LED*

Let $\mathcal{S}$ be a class of program schemes, e.g. straight-line programs, or flowcharts, or recursive procedures, over a given language L. One can add $\mathcal{S}$ as an additional parameter in the definition (3.1) of LED where in 3.1.1 schemes S and Q are assumed to range over $\mathcal{S}$. The meaning of $\equiv$ (total equivalence) remains unchanged. In this way we obtain a logic LED($\mathcal{S}$, L) which can be viewed as a fragment of LED(L) as far as $\mathcal{S}$ is translatable into ED(L). There is a tendency (cf. [30]) to treat effective definitional schemes as those of maximal computational power. Therefore for reasonable classes $\mathcal{S}$, the resulting logics LED($\mathcal{S}$) can be always viewed as fragments of LED.

Let LED_1 and LED_2 be two fragments of LED(L) for a finite language L. LED_1 is said to be *interpretable* in LED_2 if for every $\alpha \in LED_1$ there exists a $\beta \in LED_2$ such that $\models \alpha \leftrightarrow \beta$. If LED_1 is interpretable into LED_2 then we write $LED_1 \leq LED_2$. $LED_1 \approx LED_2$ means that $LED_1 \leq LED_2$ and $LED_2 \leq LED_1$. Finally, $LED_1 < LED_2$ means $LED_1 \leq LED_2$ and $LED_1 \not\approx LED_2$. LED_1 is said to be *semantically equivalent* to LED_2 if $LED_1 \approx LED_2$.

For example, LED based upon straight-line programs is semantically equivalent to first order logic with equality. LED based upon flowcharts (with equality test) is semantically equivalent to Algorithmic Logic with individual and iteration quantifiers (cf. [1] and [25] for the necessary definitions), as well as to first-order Deterministic Dynamic Logic (based on regular programs) (cf. [15]).

Just as with logics, we can compare classes of program schemes (cf. [9, 14, 23, 30]). We write $\mathcal{S}_1 \leq \mathcal{S}_2$ if $\mathcal{S}_1$ is translatable into $\mathcal{S}_2$. As above $\mathcal{S}_1 \approx \mathcal{S}_2$ means $\mathcal{S}_1 \leq \mathcal{S}_2$ and $\mathcal{S}_2 \leq \mathcal{S}_1$, and $\mathcal{S}_1 < \mathcal{S}_2$ means $\mathcal{S}_1 \leq \mathcal{S}_2$ and $\mathcal{S}_1 \not\approx \mathcal{S}_2$.

The following questions arise naturally:

1. For what classes $\mathcal{S}_1$, $\mathcal{S}_2$ of program schemes does the following implication hold:

$\mathcal{S}_1 < \mathcal{S}_2$ implies $\mathrm{LED}(\mathcal{S}_1) < \mathrm{LED}(\mathcal{S}_2)$?

2. Is it possible to find two classes of program schemes such that $\mathcal{S}_1 < \mathcal{S}_2$, flowcharts are translatable into $\mathcal{S}_1$, and $\mathrm{LED}(\mathcal{S}_1) \approx \mathrm{LED}(\mathcal{S}_2)$?

3. Is the full LED semantically equivalent to any of its fragments $\mathrm{LED}(\mathcal{S})$ with flowcharts translatable into $\mathcal{S}$ and $\mathcal{S} < \mathrm{ED}$?

5.2 *The Role of the Equality Test*

In schematology it is not usually assumed that the program schemes can always test for equality between individuals (cf. [9, 14, 30]).

Therefore the following problems seems to be worth investigating:

4. For a given class of $\mathcal{S}$ of program schemes let $\mathcal{S}_e$ denote the class of $\mathcal{S}$-schemes augmented by the equality predicate. What is the relationship between $\mathrm{LED}(\mathcal{S})$ and $\mathrm{LED}(\mathcal{S}_e)$?

5. In particular, when does $\mathrm{LED}(\mathcal{S}) \approx \mathrm{LED}(\mathcal{S}_e)$ hold?

Remark: If $\mathcal{S}$ is a class of program schemes without the equality predicate among its basic relations then $S \doteq Q$ usually cannot be replaced by the termination property of a scheme in $\mathcal{S}$. (i.e. Proposition 2.2.1(ii) fails).

5.3 *One-and-a-half-Order LED*

This extension of LED arises via quantification over schemes. Actually there are two possible extensions:

(I) LED^{+} : quantification over schemes and over individuals.

(II) LED^{*} : only quantification over schemes is allowed.

Formulas in $\mathrm{LED}^{*}(\mathcal{S})$ express various recursion-theoretic properties of $\mathcal{S}$-schemes. For example the formula $\forall S_1 \forall S_2 \exists S_3 (S_1 \doteq S_2 \leftrightarrow S_3{\downarrow})$ is a formula of LED^{*} true in ED, and the remark in 5.2 says that this formula need not to be true in $\mathcal{S}$ for a class $\mathcal{S}$ of program schemes without the equality predicate. To take another example, consider the formula $\forall S_1 \forall S_2 \exists S_3 [(S_1{\downarrow} \vee S_2{\downarrow}) \leftrightarrow S_3{\downarrow}]$. This formula is true in ED as well as in the class of all flowcharts but is *not* true in the class of all flowcharts augmented by *one* stack.

Problems

6. Given a class $\mathscr{S}$ of program schemes compare LED($\mathscr{S}$) and LED*($\mathscr{S}$).

7. Call two classes $\mathscr{S}_1$, $\mathscr{S}_2$ of program schemes *similar* if for every sentence α in LED*, α is true in $\mathscr{S}_1$ iff α is true in $\mathscr{S}_2$. Is the class of all flowcharts similar to ED?

8. Given a class $\mathscr{S}$ of program schemes, investigate the decidability of the set of all LED* sentences true in $\mathscr{S}$.

9. Given a class $\mathscr{S}$ of program schemes, investigate the axiomatizability of the set of all LED* sentences true in $\mathscr{S}$.

10. Is there a class $\mathscr{S}$ of program schemes which is uniquely definable (up to semantic inter-translatability) by LED* sentences true in $\mathscr{S}$?

5.4 *Expressive Power and Unique Definability*

11. Let $\mathscr{S}_1$ and $\mathscr{S}_2$ be classes of program schemes such that LED($\mathscr{S}_1$) < LED($\mathscr{S}_2$). Is it always possible to find a structure which is uniquely definable in LED($\mathscr{S}_2$) by a single sentence but not definable in LED($\mathscr{S}_1$) by any set of LED($\mathscr{S}_1$) sentences? by a single LED($\mathscr{S}_1$) sentence?

5.5 *Unique Definability*

12. Characterize structures which are uniquely definable by a single open LED formula over a language without constants.

13. Characterize structures which are uniquely definable by a single LED formula.

14. Characterize structures which are uniquely definable by a set of LED formulas.

15. Are there any reasonable results concerning unique definability of structures in fragments of LED (eg. for flowcharts)?

5.6 *The Hanf Number for Fragments of LED*

16. Is there a class $\mathscr{S}$ of program schemes such that LED($\mathscr{S}$) < LED, and both LED($\mathscr{S}$) and LED have the same Hanf numbers?

17. Compute Hanf numbers for various well-behaving fragments of LED (eg. for flowcharts, recursive procedures, flowcharts with counters).

5.7 *Eds-Complete Classes*

18. Let L be a finite language satisfying the following conditions: $L_F = \{f\}$, $L_R \neq \phi$, $\rho_L(f) = 1$, and for all $r \in L_R$, $\rho_L(r) = 1$. Is the class of all L-structures eds-complete?

19. Is the class of all models of *Peano arithmetic* eds-complete?

5.8 *Tools to Construct Models*

20. In this paper we have presented two results (Theorems 3.5.7 and 4.2.2) which can be viewed as tools to construct models. The first of these results was derived from the Model Existence Theorem in $L_{\omega_1\omega}$, and the second from the Omitting Types Theorem in $L_{\omega\omega}$. Are there any results specific to LED which also provide tools to construct models, and which cover 3.5.7 and 4.2.2?

This question seems to be important since in the presence of stronger tools to construct models some of the problems stated above may succumb to standard model-theoretic solutions.

REFERENCES

[1] Banachowski, L., Kreczmar, A., Mirkowska, G., Rasiowa, H. and Salwicki, A., An Introduction to Algorithmic Logic. Mathematical Investigations in the Theory of Programs. In Mazurkiewicz and Pawlak (eds) *Math. Found of Comp. Sc.* Banach Center Publications, Warsaw 1977, pp. 7-99.

[2] Bell, J. and Machover, M., *A Course in Mathematical* Logic. North-Holland Publ. Co., Amsterdam 1977.

[3] Bergstra, J. and Meyer, J. J. C. On the Quantifier Free Fragment of Logic of Effective Definitions. Leiden University Report (80-4), 1980.

[4] Bergstra, J. and Tiuryn, J., Implicit Definability of Algebraic Structures by Means of Program Properties. Abstract: in Budach (ed.) *Fundamentals of Comp. Th.* Academie-Verlag Berlin 1979. The full version will appear in *Fundamenta Informaticae*.

[5] Bergstra, J., and Tiuryn, J., Algorithmic Degrees of Algebraic Structures. Leiden University Report (79-6), 1979.

[6] Bergstra, J., Tiuryn, J., and Tucker, J.V., Correctness Theories and Program Equivalence. Mathematisch Centrum Report (IW 119/79). Amsterdam 1979.

[7] Chang, C.C. and Keisler, H.J., *Model Theory*. North-Holland Publ. Co., Amsterdam 1973.

[8] Constable, R.L., A Constructive Programming Logic. In Gilchrist (ed.) *Information Processing '77*. Proc. of IFIP Congress 77. North-Holland Publ. Co. Amsterdam 1977, pp. 733-738.

[9] Constable, R.L. and Gries, D., On Classes of Program Schemata. *SIAM Journal on Computing 1* (1972) pp. 66-118.

[10] Engeler, E., Algorithmic Logic. In de Bakker (ed.) *Mathematical Centre Tracts* (63) Amsterdam 1975, pp. 57-85.

[11] Engeler, E., Generalized Galois Theory and its Application to Complexity. Berichte des Instituts für Informatik (24) ETH Zürich (1978).

[12] Friedman, H., Algorithmic Procedures, Generalized Turing Algorithms, and Elementary Recursion Theory. In Gandy and Yates (eds) *Logic Colloquium '69*, North-Holland Publ. Co., Amsterdam 1971, pp. 361-390.

[13] Goguen, J.A., Thatcher, J.W., and Wagner, E.G., An Initial Algebra Approach to the Specification Correctness and Implementation of Abstract Data Types. In Yeh (ed.) *Current Trends in Programming Methodology*, (vol. 3) Data Structuring. Prentice-Hall 1977. Automatic Computation Series.

[14] Greibach, S.A., *Theory of Program Structures: Schemes, Semantics, Verification.* Lecture Notes in Comp. Sc. 36, Springer Verlag, Berlin 1975.

[15] Harel, D., *First-order Dynamic Logic.* Lecture Notes in Comp. Sc. 68, Springer Verlag, Berlin 1979.

[16] Keisler, H.J., *Model Theory for Infinitary Logic.* North-Holland Publ. Co., Amsterdam 1972.

[17] Kfoury, D.J., Comparing Algebraic Structures up to Algorithmic Equivalence. In Nivat (ed.) *Automata, Languages and Programming.* North-Holland Publ. Co., Amsterdam 1972, pp. 253-264.

[18] Kfoury, D.J., Translatability of schemes over restricted interpretations. *Journal of Comp. and Syst. Sc. 8* (1974) pp. 387-408.

[19] Kreczmar, A. Programmability in Fields. *Fundamenta Informaticae* I, 2. (1977) pp. 195-230.

[20] Lopez-Escobar, E., An interpolation theorem for denumerably long sentences. *Fundamenta Mathematicae* LVII, (1965) pp. 253-272.

[21] Meyer, A.R. and Grief, I., Can Partial Correctness Assertions Specify Programming Language Semantics? In Weihrauch (ed.) *Theoretical Comp Sc. 4th GI Conference.* Lecture Notes in Comp. Sc. 67, Springer Verlag, Berlin (1979) pp. 25-26.

[22] Meyer, A.R. and Halpern, J.Y. Axiomatic Definitions of Programming Languages: A Theoretical Assessment. In *Proceedings of the 7th Annual Symposium on the Principles of Programming Languages.* pp. 203-212, January 1980.

[23] Moldestad, J., Stoltenberg-Hansen, V. and Tucker, J.V., Finite Algorithmic Procedures and Computation Theories. To appear in *Mathematica Scandanavia.*

[24] Parikh, R., The Completeness of Propositional Dynamic Logic. In Winkowski (ed.) *Proc. of MFCS '78.* Lecture Notes in Comp. Sc. 64, Springer Verlag, Berlin 1978, pp. 403-415.

[25] Rasiowa, H., Algorithmic Logic. ICSPAS Report 281, Warsaw 1977.

[26] Rasiowa, H., ω^+-valued Algorithmic Logic As a Tool To Investigate Procedures. In Blikle (ed.) *Proc. of MFCS '74* Lecture Notes in Comp. Sc. 28, Springer Verlag, Berlin 1974, pp. 423-450.

[27] Rasiowa, H., Logic of Complex Algorithms. In Budach (ed.) *Fundamentals of Comp. Th.* Akademie Verlag, Berlin 1979. pp. 371-380.

[28] Rogers, H., *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York 1967.

[29] Salwicki, A., On Algorithmic Theory of Stacks. In Winkowski (ed.) *Proc. of MFCS '78.* Lecture Notes in Comp. Sc. 64. Springer Verlag, Berlin 1978.

[30] Shepherdson, J.C., Computation Over Abstract Structures: Serial and Parallel Procedures and Friedman's Effective Definitional Schemes. In Shepherdson and Rose (eds.) *Logic Colloquium '73.* North-Holland, Amsterdam 1973, pp. 445-513.

[31] Tiuryn, J., Algebraic Aspects of Logic Based on Term Algebra of r.e. trees. Warsaw University Report 1978.

[32] Tiuryn, J., Completeness Theorem for Logic of Effective Definitions. To appear in *Proc. Coll. Logic in Programming*, Salgotarjan 1978, North-Holland Publishing Co.

[33] Tiuryn, J., Logic of Effective Definitions. R.W.T.H.-Aachen Report 55, 1979. To appear in *Fundamenta Informaticae.*

[34] Urzyczyn, P., On Algorithmically Trivial Structures. Warsaw University Report 1979.